ADT PLM

Programmer's Learning Machine

Matthieu Nicolas

IJD Seminar, 2016-02-02

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Presentation of PLM

Purposes

- Application to learn programming.

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.

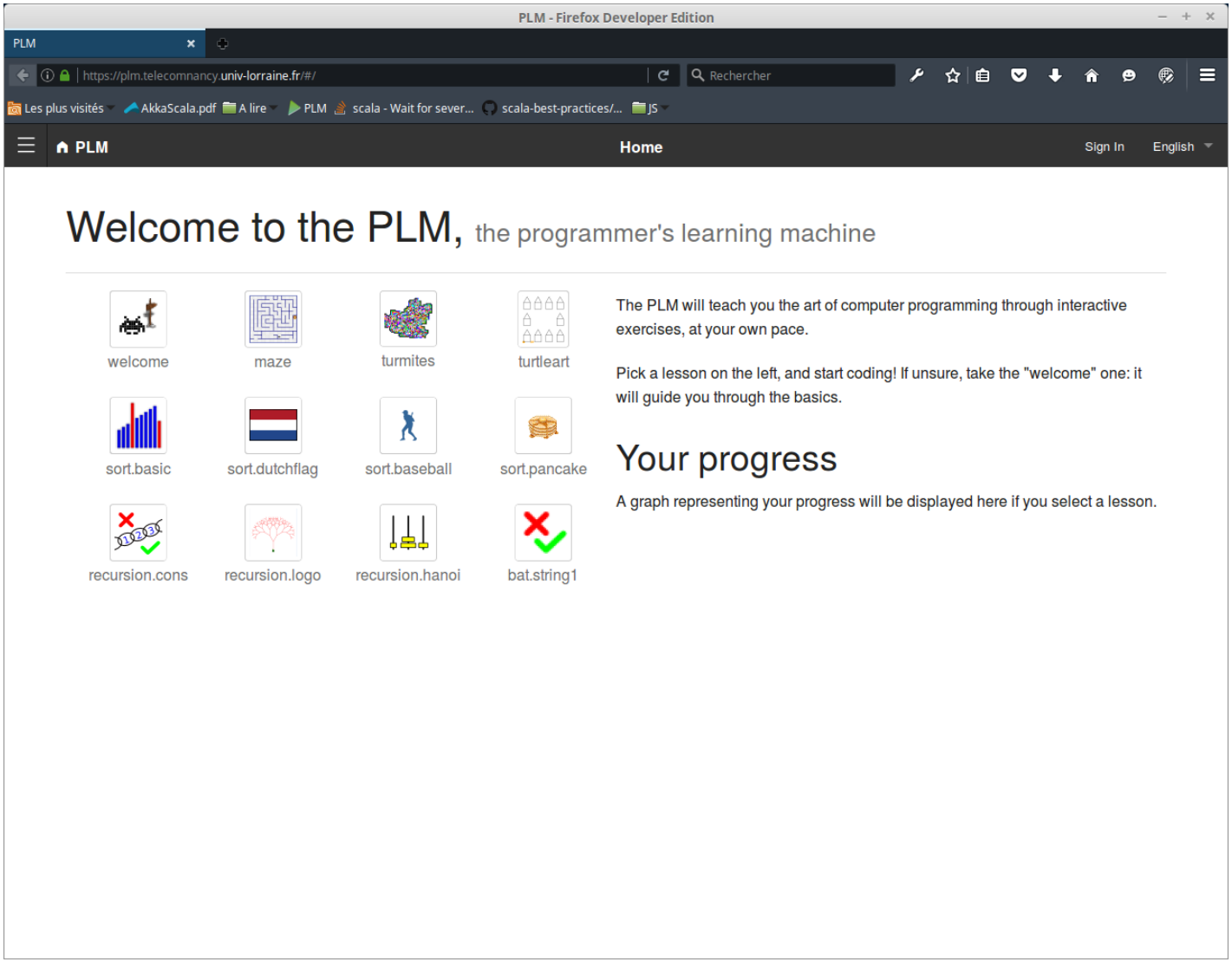
Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.
- Used at TELECOM Nancy since 2008.

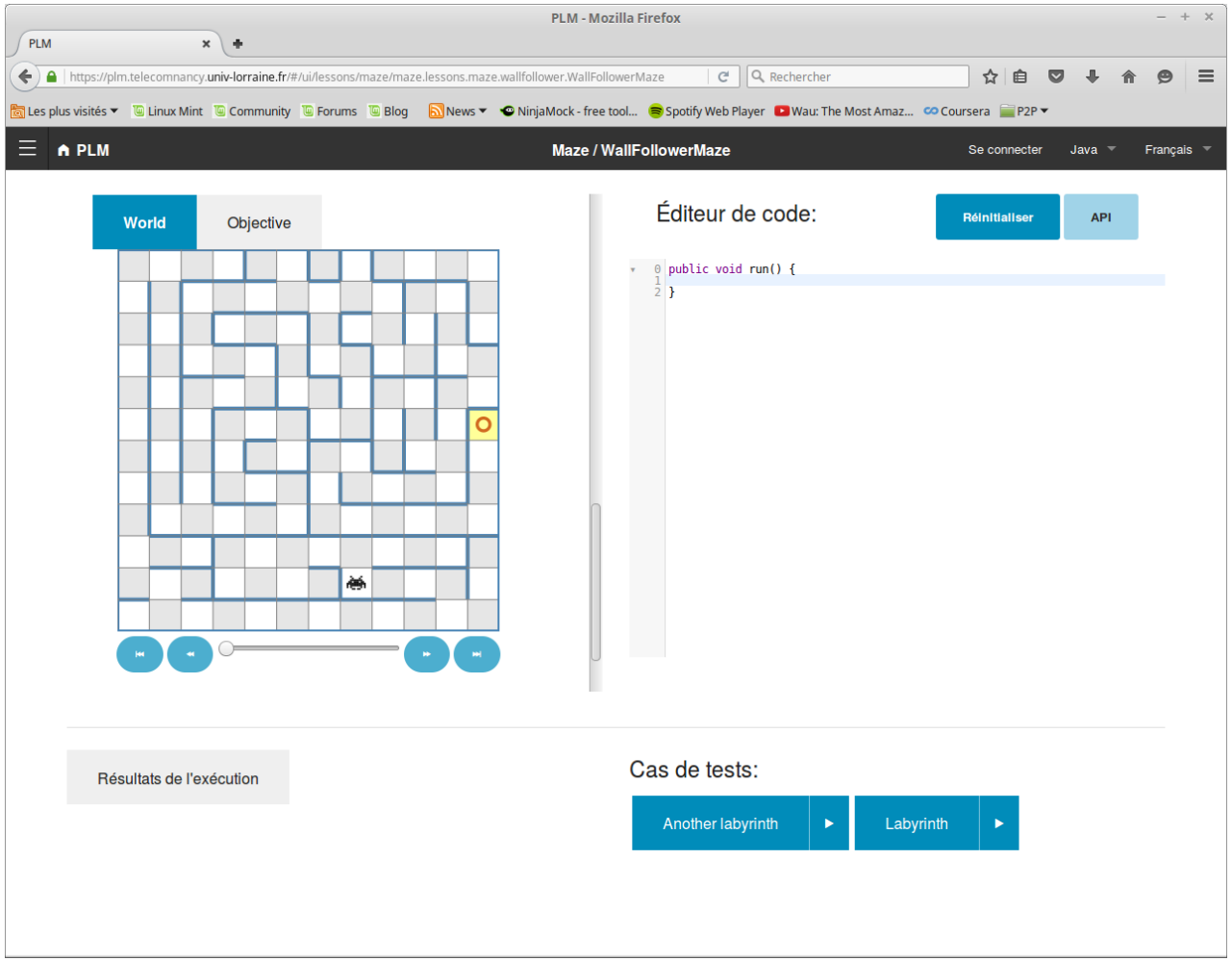
Presentation of PLM

Quick demo



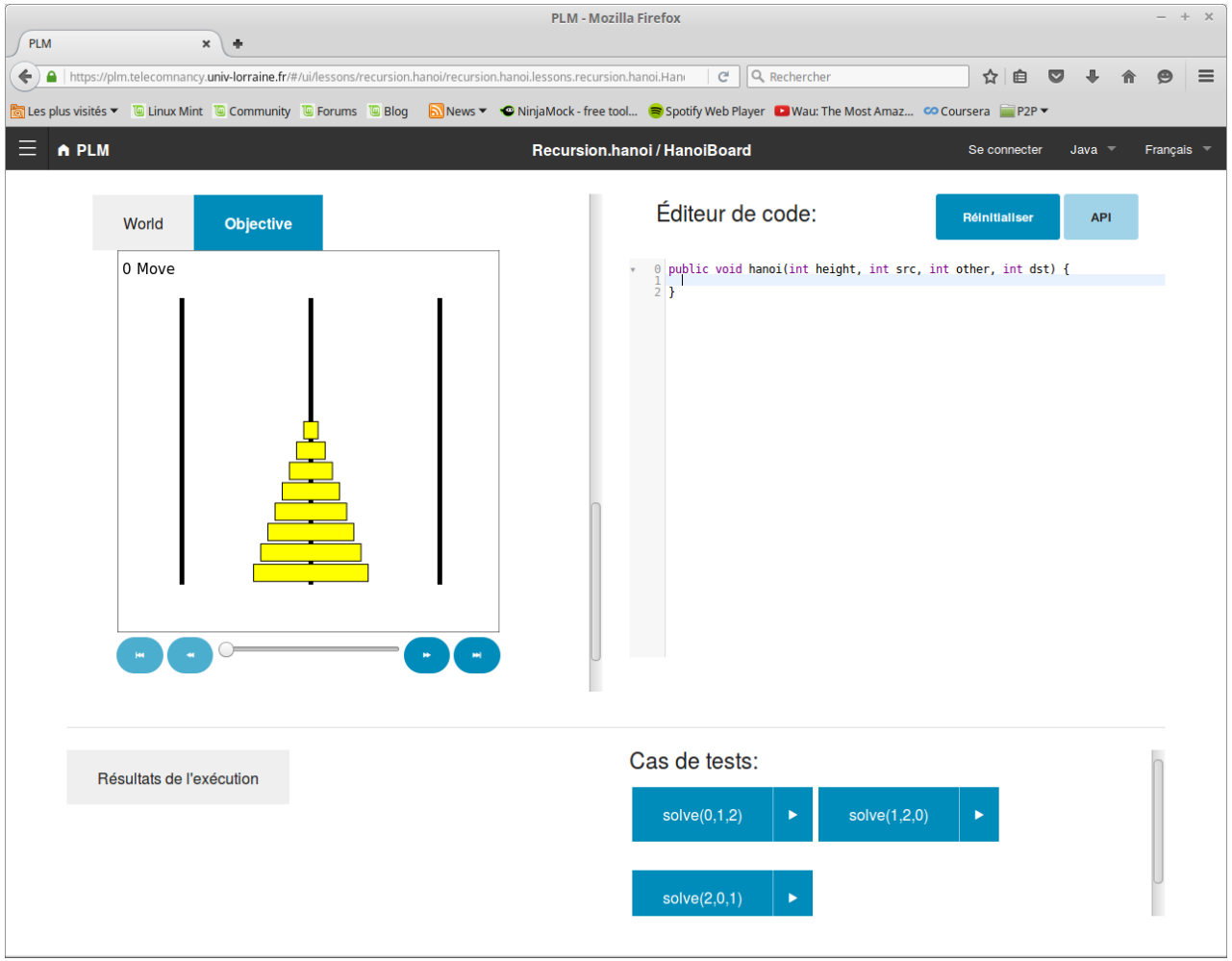
Presentation of PLM

12 lessons, 200 exercises



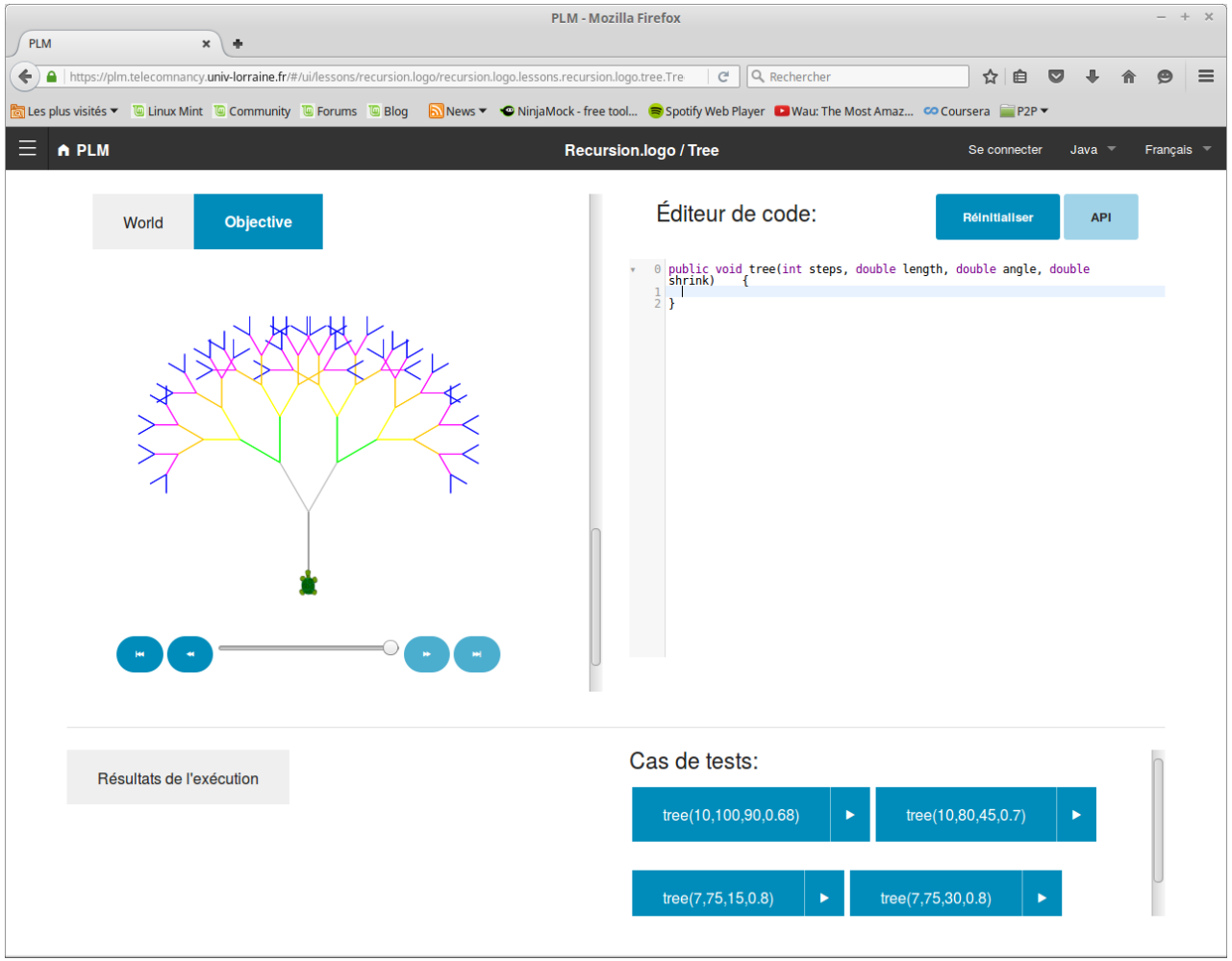
Presentation of PLM

12 lessons, 200 exercises



Presentation of PLM

12 lessons, 200 exercises



Presentation of PLM

Supported languages

- English
- French
- Brazilian Portuguese

Presentation of PLM

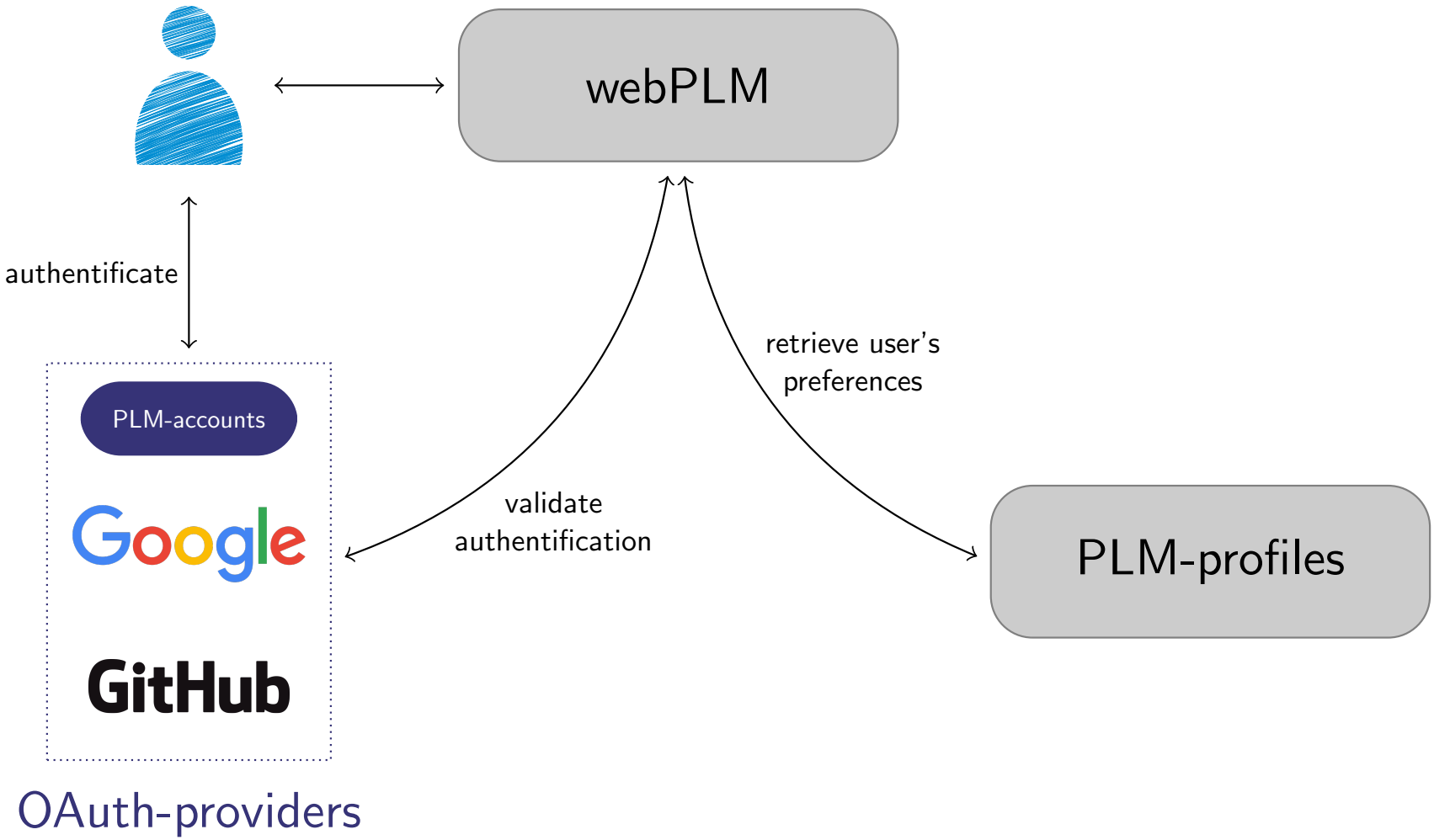
Supported programming languages

- Formerly a fat client
  - Written in Java

- Formerly a fat client
  - Written in Java
- Switch to a web application
  - Server implemented in Scala using PlayFramework
  - User interface written in Javascript using AngularJS and Foundation

Presentation of PLM

Application's architecture

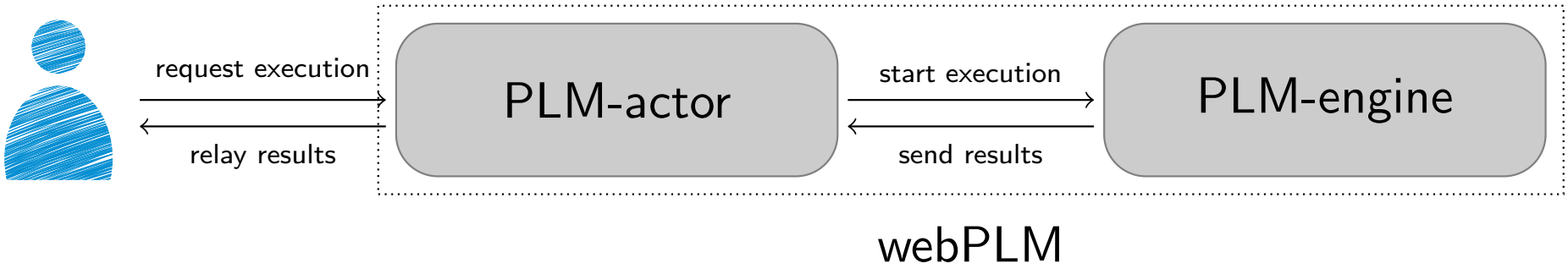


Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

User's code's assessment

Execution components



User's code's assessment

Limits

- Run on the same machine, same JVM

User's code's assessment

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops

User's code's assessment

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Storage jamming with files

User's code's assessment

Limits

- Run on the same machine, same JVM

- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Storage jamming with files
- And from *System.exit(whatever)*?

User's code's assessment

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Storage jamming with files
- And from *System.exit(whatever)*?
- Scalability issues

User's code's assessment

Chosen solution

- Delegate the execution to workers
  - Called *Judges* in the litterature
  - Execute user's code and send back result to webPLM

User's code's assessment

Chosen solution

- Delegate the execution to workers
  - Called *Judges* in the litterature
  - Execute user's code and send back result to webPLM
- *Let it crash* strategy
  - Prevent obvious issues with a security manager
  - Handle timeout and crash

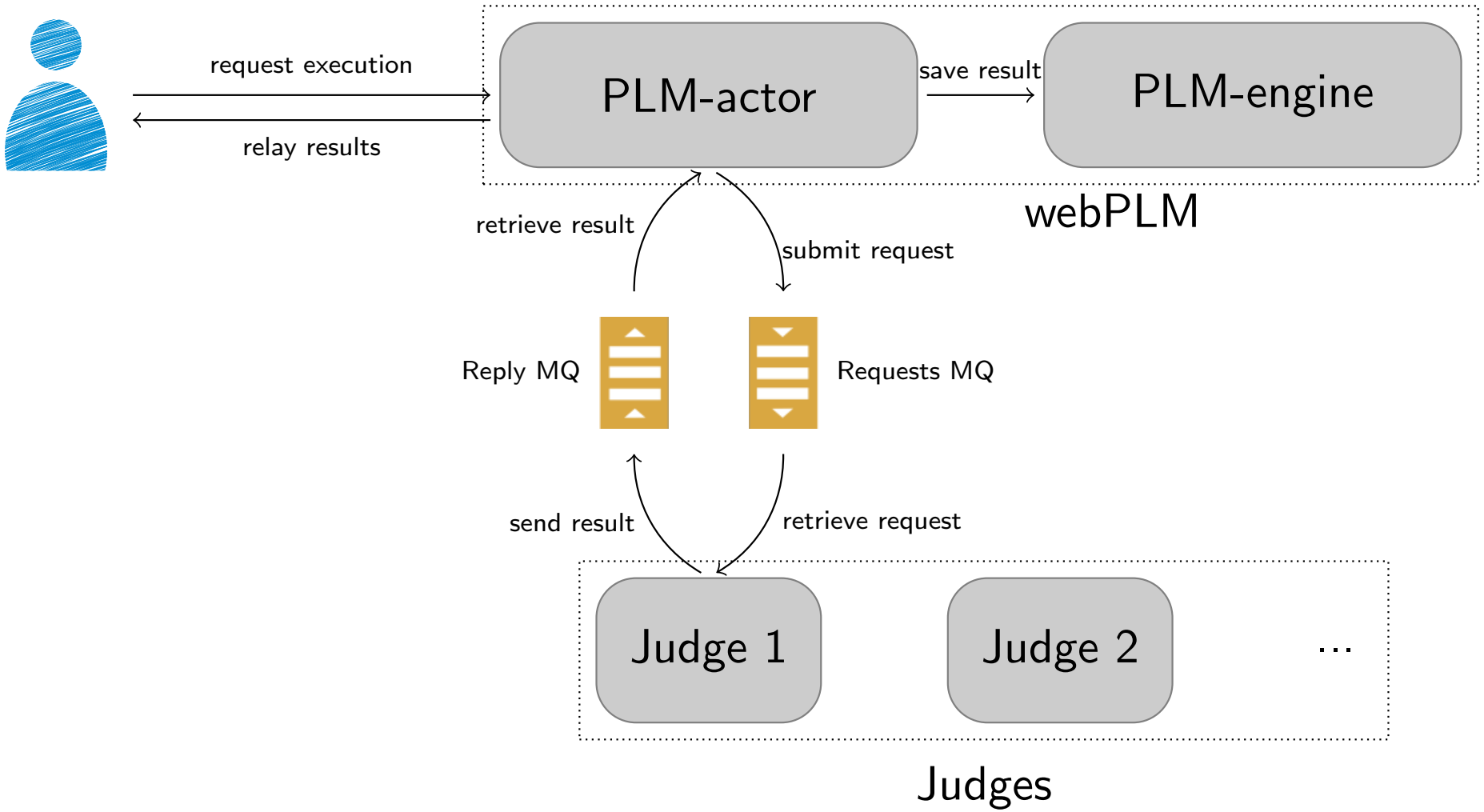
User's code's assessment

Chosen solution

- Delegate the execution to workers
  - Called *Judges* in the litterature
  - Execute user's code and send back result to webPLM
- *Let it crash* strategy
  - Prevent obvious issues with a security manager
  - Handle timeout and crash
- Distribute workload using message queues
  - One queue for requests
  - One queue per result

User's code's assessment

Architecture with judges



User's code's assessment

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements

User's code's assessment

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements
- Cons:
  - Need to deploy them easily
  - Should be able to reset them
  - Have to restrict their resources usage

User's code's assessment

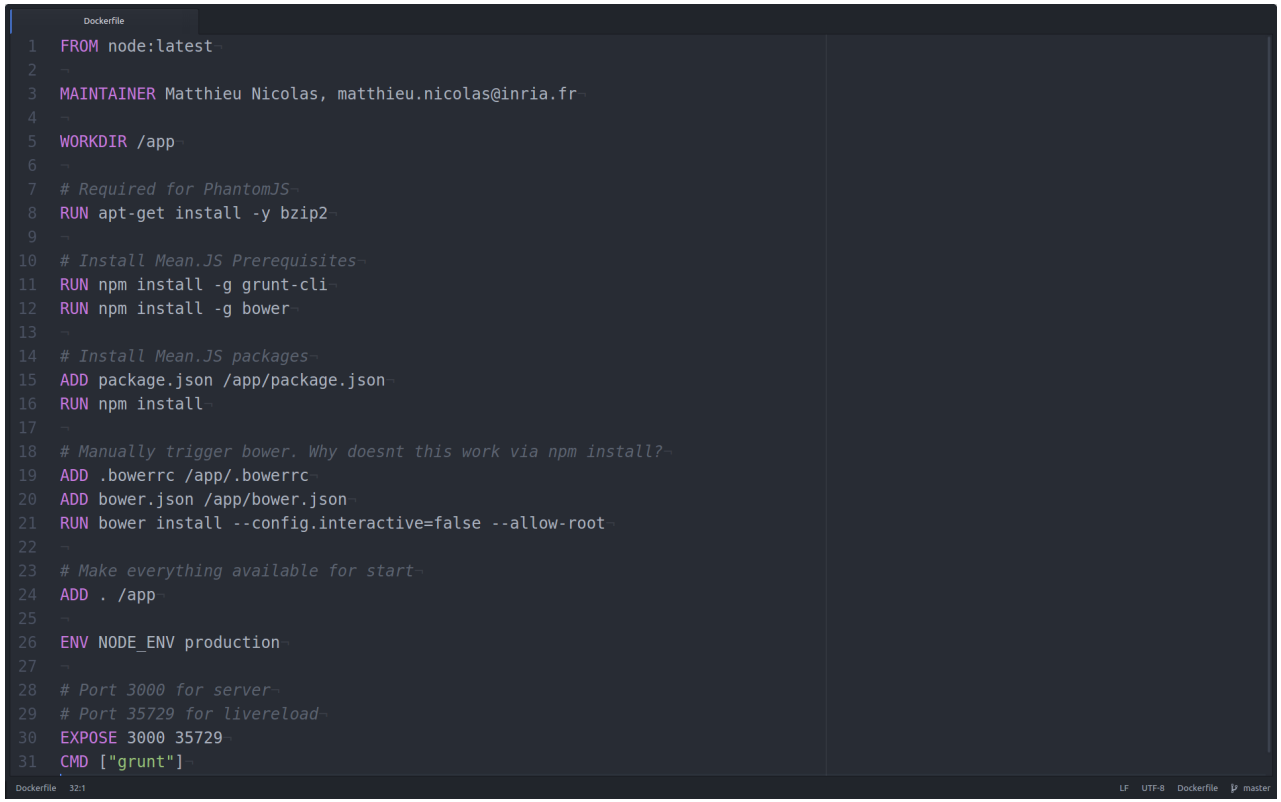
Docker

- Lightweight virtualization tool
- Build image of your application
- Run containers based on images

User's code's assessment

Example of Dockerfile

- Dockerfiles describe how to set up the application



```
Dockerfile
1  FROM node:latest
2
3  MAINTAINER Matthieu Nicolas, matthieu.nicolas@inria.fr
4
5  WORKDIR /app
6
7  # Required for PhantomJS
8  RUN apt-get install -y bzip2
9
10 # Install Mean.JS Prerequisites
11 RUN npm install -g grunt-cli
12 RUN npm install -g bower
13
14 # Install Mean.JS packages
15 ADD package.json /app/package.json
16 RUN npm install
17
18 # Manually trigger bower. Why doesnt this work via npm install?
19 ADD .bowerrc /app/.bowerrc
20 ADD bower.json /app/bower.json
21 RUN bower install --config.interactive=false --allow-root
22
23 # Make everything available for start
24 ADD . /app
25
26 ENV NODE_ENV production
27
28 # Port 3000 for server
29 # Port 35729 for livereload
30 EXPOSE 3000 35729
31 CMD ["grunt"]
Dockerfile 32:1                                LF UTF-8 Dockerfile master
```

- Run *docker build -t tag /path/to/Dockerfile* to build the image
- Start containers with *docker run tag*

User's code's assessment

More about *docker run*

- Can also manage
  - Ports

User's code's assessment

More about *docker run*

- Can also manage
  - Ports
  - Volumes

User's code's assessment

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers

User's code's assessment

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers
  - Environment variables
  - Runtime constraints on resources
  - Restart policies
  - And a **lot more**

User's code's assessment

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers
  - Environment variables
  - Runtime constraints on resources
  - Restart policies
  - And a **lot more**

- Commands can become quite complex

docker run -p 443:9443 -link plm-accounts:accounts -v ~/webPLM/logs/:/app/webplm-dist/logs webPLM

User's code's assessment

Docker-compose

- Tool to easily deploy multi-containers applications

```yaml
nginx:
  image: nginx
  ports:
    - "80:80"

messagequeue:
  image: rabbitmq:3-management
  ports:
    - "5672:5672"
    - "15672:15672"

plm:
  image: webplm
  volumes:
    - "~/.plm:/root/.plm"
    - "~/webPLM/logs/:/app/webplm-dist/logs"
  ports:
    - "443:9443"
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

profiles:
  image: plm-profiles
  ports:
    - "8080:3000"
  links:
    - db

db:
  image: mongo
  volumes:
    - "~/mongodb/data:/data/db"
  ports:
    - "27017:27017"
```

- Deploy environment with *docker-compose up*

User's code's assessment

Docker in our case

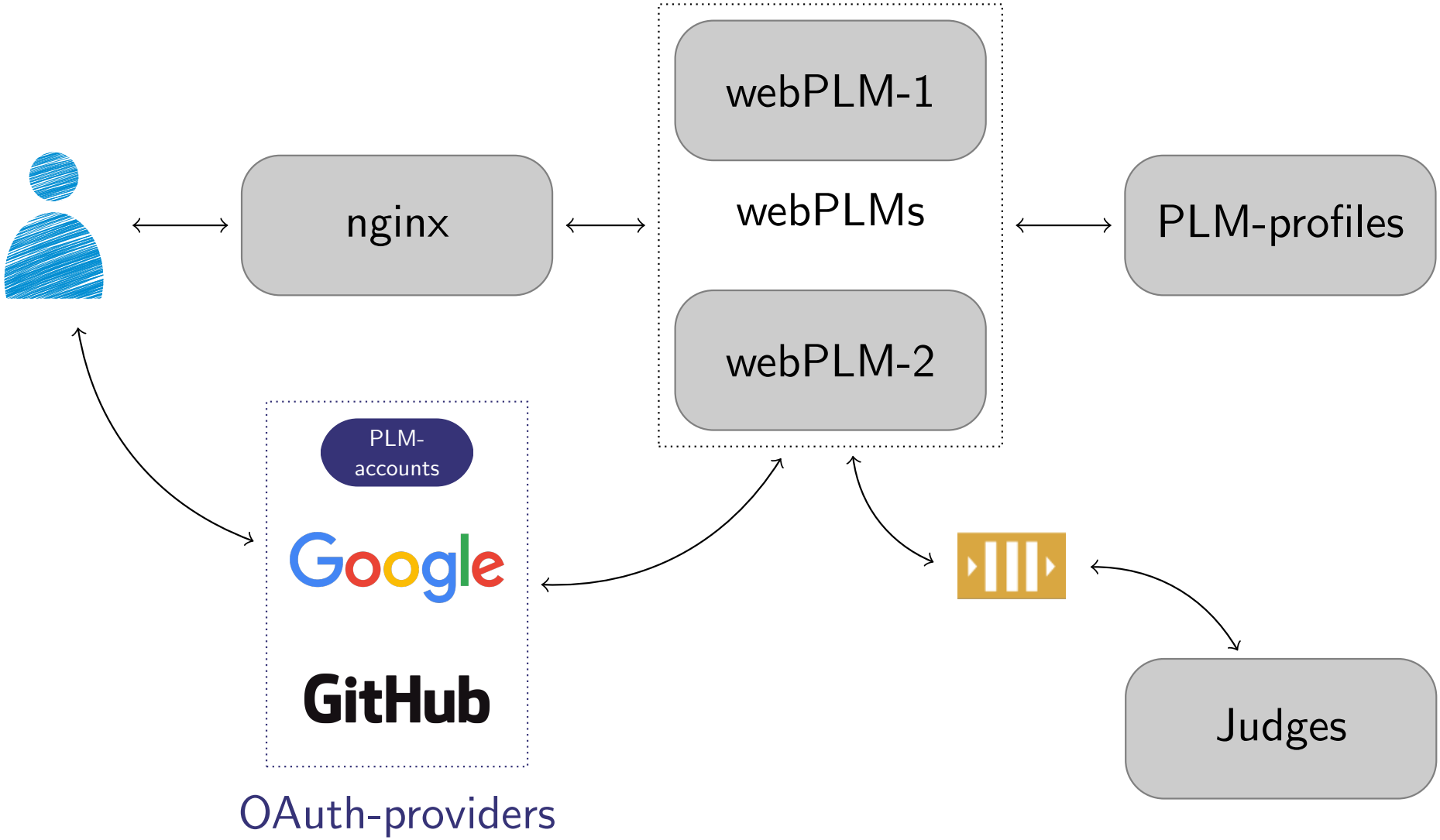
- Deploy easily all components
- Restart judges automatically
- Limit users' mischiefs

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Result

Current architecture



Result

Live-session in TELECOM Nancy

- 30 hours of live testing with 100 students.

Result

Live-session in TELECOM Nancy

- 30 hours of live testing with 100 students.
- Engine is (almost) working fine...

Result

Live-session in TELECOM Nancy

- 30 hours of live testing with 100 students.
- Engine is (almost) working fine...
- ... but user experience needs to be improved!

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.
- No tools for monitoring set up...

Result

Live-session in TELECOM Nancy

- Can't cope with the workload.
- No tools for monitoring set up...
- ... so the bottleneck is unknown.

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. User's code's assessment
   - Challenges
   - Extraction of the execution component
   - Docker
3. Result
4. Next steps

Next steps

Refactor the code

- Rushed to release a stable version before September...
- Needed to refactor some parts of the code.
- Standardized behavior of local and server mode.

Next steps

Simplify workflow to adapt the content

- Store most content inside PLM.
- Heavy and error prone workflow.
- Need to extract the content from PLM's jar.
- Allow to implement an exercise editor.

Next steps

Solve performance issues

- Set up some monitoring tools.
- Perform some load testing to identify the bottleneck.

Questions

Thanks for your attention, any questions?